**PRACTICAL 01**

**Aim: Installing and setting enviornment Variables for working with Apache Hoodoop.**

[cloudera@quickstart ~]$ hdfs dfs -ls /

Found 6 items

drwxrwxrwx - hdfs supergroup 0 2017-10-23 10:29 /benchmarks

drwxr-xr-x - hbase supergroup 0 2023-01-05 21:39 /hbase

drwxr-xr-x - solr solr 0 2017-10-23 10:32 /solr

drwxrwxrwt - hdfs supergroup 0 2023-01-05 21:29 /tmp

drwxr-xr-x - hdfs supergroup 0 2017-10-23 10:31 /user

drwxr-xr-x - hdfs supergroup 0 2017-10-23 10:31 /var

[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -mkdir/inputdirectory

-mkdir/inputdirectory: Unknown command

[cloudera@quickstart ~]$

[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -mkdir /inputdirectory

[cloudera@quickstart ~]$ hdfs dfs -ls /

Found 7 items

drwxrwxrwx - hdfs supergroup 0 2017-10-23 10:29 /benchmarks

drwxr-xr-x - hbase supergroup 0 2023-01-05 21:39 /hbase

drwxr-xr-x - hdfs supergroup 0 2023-01-05 22:42 /inputdirectory

drwxr-xr-x - solr solr 0 2017-10-23 10:32 /solr

drwxrwxrwt - hdfs supergroup 0 2023-01-05 21:29 /tmp

drwxr-xr-x - hdfs supergroup 0 2017-10-23 10:31 /user

drwxr-xr-x - hdfs supergroup 0 2017-10-23 10:31 /var

[cloudera@quickstart ~]$ cat>/home/cloudera/processfile.txt

Hii How are u Hii i am fine

^Z

[1]+ Stopped cat > /home/cloudera/processfile.txt

[cloudera@quickstart ~]$ cat /home/cloudera/processfile.txt

Hii How are u Hii i am fine

[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -chmod -R 777/inputdirectory

-chmod: Not enough arguments: expected 2 but got 1

Usage: hadoop fs [generic options] -chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...

[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -put /home/cloudera/processfile.txt /inputdirectory

[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -put /home/cloudera/processfile.txt /inputdirectory

put: `/inputdirectory/processfile.txt': File exists

[cloudera@quickstart ~]$ cat /home/cloudera/processfile.txt

Hii How are u Hii i am fine

[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -chmod -R 777/inputdirectory

-chmod: Not enough arguments: expected 2 but got 1

Usage: hadoop fs [generic options] -chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...

[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -put /home/cloudera/processfile.txt/inputdirectory

put: `.': No such file or directory

[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -put /home/cloudera/processfile.txt /inputdirectory

put: `/inputdirectory/processfile.txt': File exists

[cloudera@quickstart ~]$ hdfs dfs -ls /inputdirectory

Found 1 items

-rw-r--r-- 1 hdfs supergroup 28 2023-01-05 22:47 /inputdirectory/processfile.txt

[cloudera@quickstart ~]$ hadoop jar /home/cloudera/WordCount.jar WordCount /inputdirectory/processfile.txt /out1

23/01/05 23:00:40 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032

23/01/05 23:00:43 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.

23/01/05 23:00:44 INFO input.FileInputFormat: Total input paths to process : 1

23/01/05 23:00:44 INFO mapreduce.JobSubmitter: number of splits:1

23/01/05 23:00:45 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1672983532404\_0001

23/01/05 23:00:50 INFO impl.YarnClientImpl: Submitted application application\_1672983532404\_0001

23/01/05 23:00:51 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application\_1672983532404\_0001/

23/01/05 23:00:51 INFO mapreduce.Job: Running job: job\_1672983532404\_0001

23/01/05 23:01:39 INFO mapreduce.Job: Job job\_1672983532404\_0001 running in uber mode : false

23/01/05 23:01:39 INFO mapreduce.Job: map 0% reduce 0%

23/01/05 23:02:11 INFO mapreduce.Job: map 100% reduce 0%

23/01/05 23:02:20 INFO mapreduce.Job: map 100% reduce 100%

23/01/05 23:02:20 INFO mapreduce.Job: Job job\_1672983532404\_0001 completed successfully

23/01/05 23:02:20 INFO mapreduce.Job: Counters: 49

File System Counters

FILE: Number of bytes read=72

FILE: Number of bytes written=286861

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=155

HDFS: Number of bytes written=38

HDFS: Number of read operations=6

HDFS: Number of large read operations=0

HDFS: Number of write operations=2

Job Counters

Launched map tasks=1

Launched reduce tasks=1

Data-local map tasks=1

Total time spent by all maps in occupied slots (ms)=29049

Total time spent by all reduces in occupied slots (ms)=5729

Total time spent by all map tasks (ms)=29049

Total time spent by all reduce tasks (ms)=5729

Total vcore-milliseconds taken by all map tasks=29049

Total vcore-milliseconds taken by all reduce tasks=5729

Total megabyte-milliseconds taken by all map tasks=29746176

Total megabyte-milliseconds taken by all reduce tasks=5866496

Map-Reduce Framework

Map input records=1

Map output records=8

Map output bytes=60

Map output materialized bytes=72

Input split bytes=127

Combine input records=8

Combine output records=7

Reduce input groups=7

Reduce shuffle bytes=72

Reduce input records=7

Reduce output records=7

Spilled Records=14

Shuffled Maps =1

Failed Shuffles=0

Merged Map outputs=1

GC time elapsed (ms)=791

CPU time spent (ms)=3270

Physical memory (bytes) snapshot=459796480

Virtual memory (bytes) snapshot=3146891264

Total committed heap usage (bytes)=389021696

Shuffle Errors

BAD\_ID=0

CONNECTION=0

IO\_ERROR=0

WRONG\_LENGTH=0

WRONG\_MAP=0

WRONG\_REDUCE=0

File Input Format Counters

Bytes Read=28

File Output Format Counters

Bytes Written=38

[cloudera@quickstart ~]$ hdfs dfs -ls /out1

Found 2 items

-rw-r--r-- 1 cloudera supergroup 0 2023-01-05 23:02 /out1/\_SUCCESS

-rw-r--r-- 1 cloudera supergroup 38 2023-01-05 23:02 /out1/part-r-00000

[cloudera@quickstart ~]$ hdfs dfs -cat /out1/part -r-00000

cat: `/out1/part': No such file or directory

cat: `-r-00000': No such file or directory

[cloudera@quickstart ~]$ hdfs dfs -cat /out1/part-r-00000

Hii 2

How 1

am 1

are 1

fine 1

i 1

u 1

[cloudera@quickstart ~]$

**PRACTICAL 02**

**Aim: Implementing Map Reduce Program for Word Count Problem**

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

public static class TokenizerMapper

extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

public void map(Object key, Text value, Context context

) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

word.set(itr.nextToken());

context.write(word, one);

}

}

}

public static class IntSumReducer

extends Reducer<Text,IntWritable,Text,IntWritable> {

private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,

Context context

) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "word count");

job.setJarByClass(WordCount.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**PRACTICAL 03**

**Aim:Write a pig Script for solving counting problem.**

**Steps :**

cat> /home/cloudera/input.csv

cat /home/cloudera/input.csv

pig -x local

lines = load '/home/cloudera/input.csv' as (line:chararray);

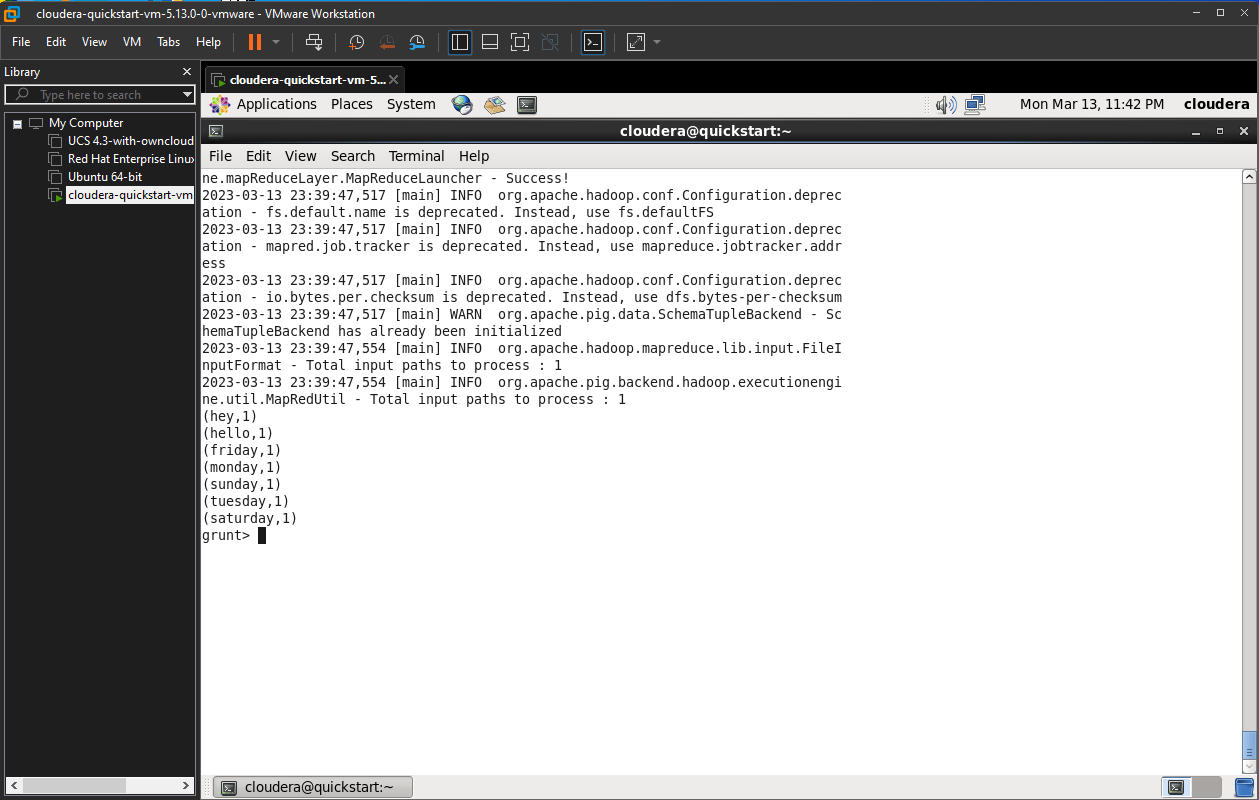
words = foreach lines GENERATE FLATTEN(TOKENIZE(line)) as woed;

grouped = GROUP words by woed;

wordcount = foreach grouped GENERATE group, COUNT(words);

dump wordcount;

OutPut:



**PRACTICAL 04**

**Aim: Install HBase and use the HBase Data model Store and retrieve data.**

Steps :

//Start HBase

hbase shell

//HBase Commands

status

version,

table\_help

whoami

//Data Definition Language

create ‘employee’, ’Name’, ’ID’, ’Designation’, ’Salary’, ’Department’

//Verify created table

list

//Disable single table

disable ‘employee’

scan ‘employee’

//or

is\_disable ‘employee’

//Disable multiple tables

disable\_all ‘e.\*’

// Enabling table

enable‘employee’

//Or

is\_enabled'employee'

//create new table

create‘student’, ‘name’, ‘age’, ‘course’

put ‘student’, ‘sharath’, ‘name:fullname’, ‘sharathkumar’

put ‘student’, ‘sharath’, ‘age:presentage’, ‘24’

put ‘student’, ‘sharath’, ‘course:pursuing’, ‘Hadoop’

put ‘student’, ‘shashank’, ‘name:fullname’, ‘shashank R

put ‘student’, ‘shashank’, ‘age:presentage’, ‘23’

put ‘student’, ‘shashank’, ‘course:pursuing’, ‘Java’

//Get Information

get ‘student’, ‘shashank’

get ‘student’, ‘sharath’

get ‘student’, ‘sharath’, ‘course’

get ‘student’, ‘shashank’, ‘course’

get ‘student’, ‘sharath’, ‘name’

//Scan

scan ‘student’

//Count

Count ‘student’

//Alter

alter ‘student’, NAME=>’name’, VERSIONS=>5

put ‘student’, ‘shashank’, ‘name:fullname’, ‘shashank Rao’

scan ‘student’

//Delete

delete ‘student’, ‘shashank’, ‘name:fullname’

**PRACTICAL 05**

**Aim: Install Hive and use Hive Create and store structured databases.**

Steps :

cat > /home/cloudera/employee.txt

1~Sachine~Pune~Product Engineering~100000~Big Data

2~Gaurav~Banglore~Sales~90000~CRM

3~Manish~Chennai~Recruiter~125000~HR

4~Bhushan~Hyderabad~Developer~50000~BFSI

cat /home/cloudera/employee.txt

sudo -u hdfs hadoop fs -put /home/cloudera/employee.txt /inputdirectroy

hdfs dfs -ls /

hdfs dfs -ls /inputdirectory

hadoop fs -cat /inputdirectory/employee.txt

hive

show databases;

create database organization;

show databases;

use organization;

show tables;

hive> create table employee(

> id int,

> name string,

> city string,

> department string,

> salary int,

> domain string)

> row format delimited

> fields terminated by '~';

show tables;

select \* from employee;

show tables;

load data inpath '/inputdirectory/employee.txt' overwrite into table employee;

show tables;

select \* from employee;

**PRACTICAL 06**

**Aim: Write a program to construct different types of K-shingles for a given document.**

Code:

install.packages("tm")

require("tm")

install.packages("devtools")

readinteger<-function()

{

n<-readline(prompt="enter value of k-1:")

k<-as.integer👎

u1<-readLines("E:/MSC Notes/file.txt")

shingle<-0

i<-0

while(i<nchar(u1)-k+1){

shingle[i]<-substr(u1,start=i,stop=i+k)

print(shingle[i])

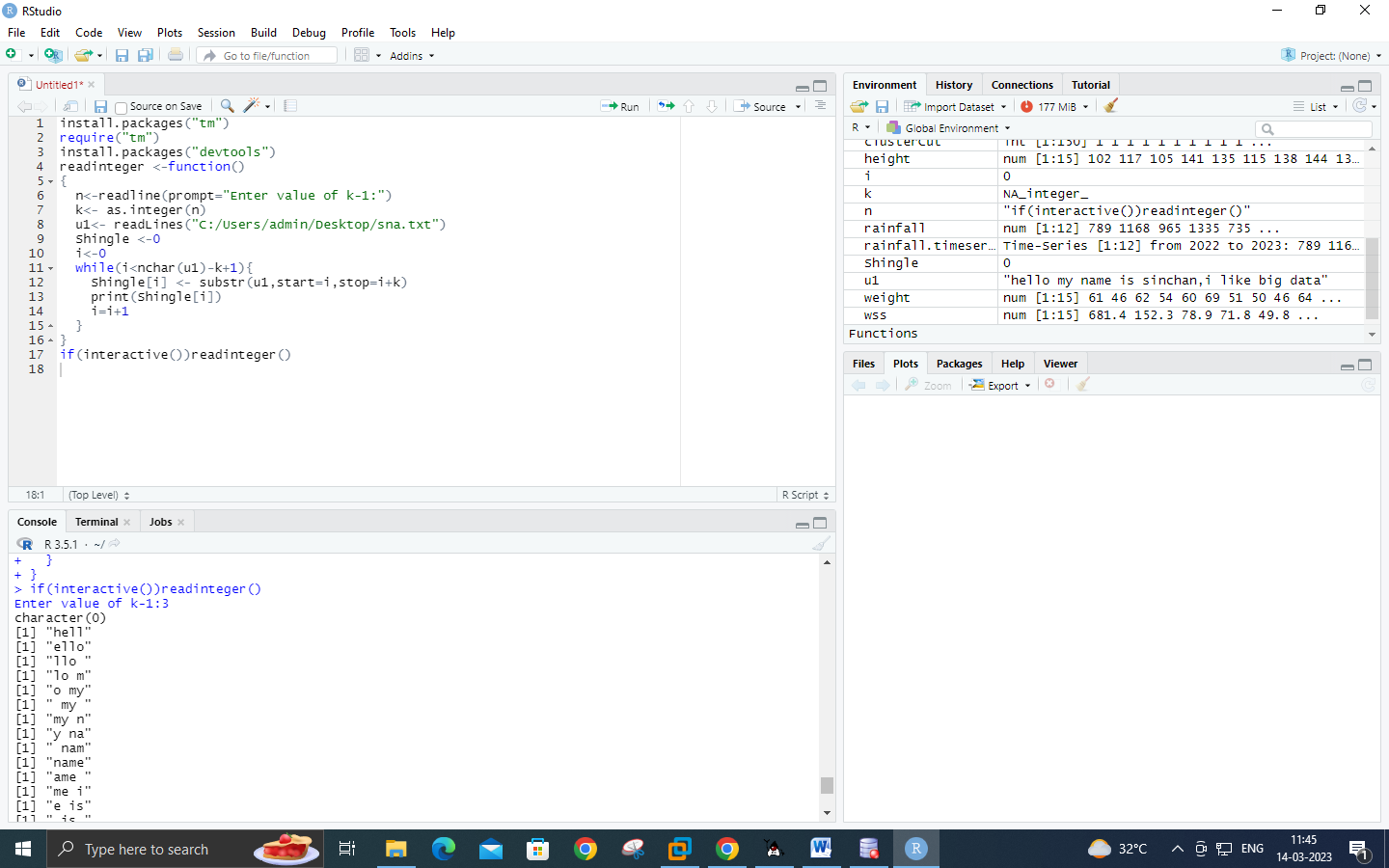
i=i+1

}

}

if(interactive())readinteger()

Output:



**PRACTICAL 07**

**Aim: Write a program to measure the similarity among the documents and detecting passages which have been reused.**

Code:

install.packages("tm")

require("tm")

install.packages("devtools")

my.corpus <- Corpus(DirSource("C:/MSC Notes/r-corpus"))

my.corpus<- tm\_map(my.corpus, removeWords ,stopwords("english"))

my.tdm<- TermDocumentMatrix(my.corpus)

my.dtm<- DocumentTermMatrix(my.corpus,control=list(weighting= weightTfIdf ,stopwords=TRUE))

my.df<- as.data.frame(inspect(my.tdm))

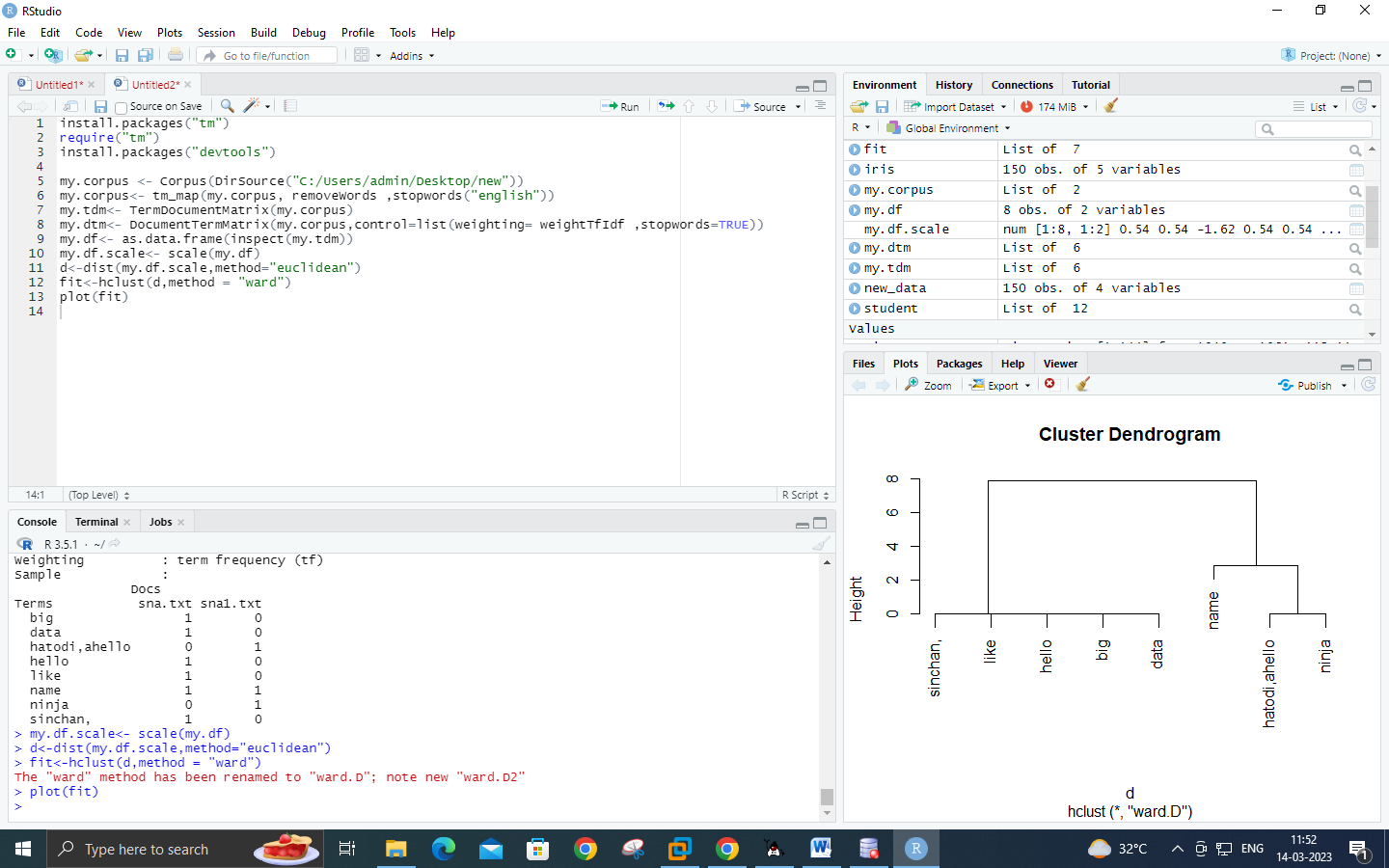
my.df.scale<- scale(my.df)

d<-dist(my.df.scale,method="euclidean")

fit<-hclust(d,method = "ward")

plot(fit)

Output:



**PRACTICAL 08**

**Aim: Write a program to compute the n-moment for a given stream where n is given**

Code:

import java.io.\*;

import java.util.\*;

public class n\_moment

{

public static void main(String args[]) {

int n=15;

String stream[]= {"a","b","c","b","d","a","c","d","a","b","d","c","a","a","b"};

int zero\_moment=0,first\_moment=0,second\_moment=0,count=1,flag=0;

ArrayList<Integer> arrlist=new ArrayList();

System.out.println("Arraylist elements are::");

for (int i=0;i<15;i++)

{

System.out.println(stream[i]+" ");

}

Arrays.sort(stream);

for(int i=1;i<n;i++)

{

if(stream[i]==stream[i-1])

{

count++;

}

else

{

//System.out.println("Hello"+i);

arrlist.add(count);

count=1;

}

}

arrlist.add(count);

zero\_moment=arrlist.size();

System.out.println("\n\n\nValue of Zeroth moment for given stream::"+zero\_moment);

for(int i=0;i<arrlist.size();i++)

{

first\_moment+=arrlist.get(i);

}

System.out.println("\n\nValue of First moment for given stream::"+first\_moment);

for (int i=0;i<arrlist.size();i++)

{

int j=arrlist.get(i);

second\_moment+=(j\*j);

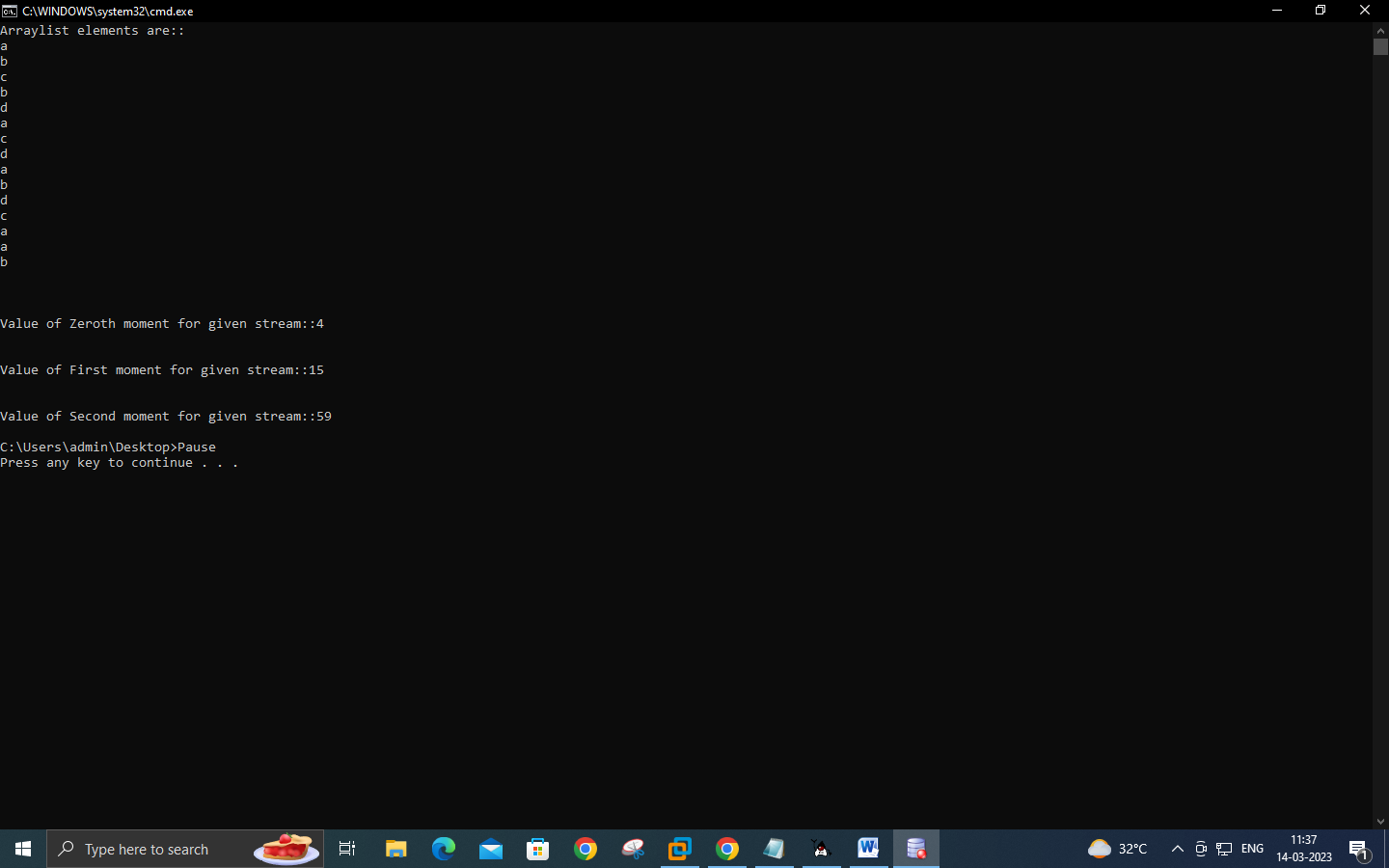
}

System.out.println("\n\nValue of Second moment for given stream::"+second\_moment);

}

}

Output:



**PRACTICAL 09**

***Aim: Write a program to demonstrate AMS Algorithm for second moments.***

Code:

import java.io.\*;

import java.util.\*;

class AMSA

{

public static int findCharCount(String stream,char XE,int random,int n)

{

int countoccurance=0;

for(int i=random;i<n;i++)

{

if(stream.charAt(i)==XE)

{

countoccurance++;

}

}

return countoccurance;

}

public static int estimateValue(int XV1,int n)

{

int ExpValue;

ExpValue=n\*(2\*XV1-1);

return ExpValue;

}

public static void main(String args[])

{

int n=15;

String stream="abcbdacdabdcaab";

int random1=3,random2=8,random3=13;

char XE1,XE2,XE3;

int XV1,XV2,XV3;

int ExpValuXE1,ExpValuXE2,ExpValuXE3;

int apprSecondMomentValue;

XE1=stream.charAt(random1-1);

XE2=stream.charAt(random2-1);

XE3=stream.charAt(random3-1);

XV1=findCharCount(stream,XE1,random1-1,n);

XV2=findCharCount(stream,XE2,random2-1,n);

XV3=findCharCount(stream,XE3,random3-1,n);

System.out.println(XE1+"="+XV1+" "+XE2+"="+XV2+" "+XE3+"="+XV3);

ExpValuXE1=estimateValue(XV1,n);

ExpValuXE2=estimateValue(XV2,n);

ExpValuXE3=estimateValue(XV3,n);

System.out.println("Expected value for"+XE1+" is::"+ExpValuXE1);

System.out.println("Expected value for"+XE2+" is::"+ExpValuXE2);

System.out.println("Expected value for"+XE3+" is::"+ExpValuXE3);

apprSecondMomentValue=(ExpValuXE1+ExpValuXE2+ExpValuXE3)/3;

System.out.println("approximate second moment value using alon-matis-szegedy is::"+apprSecondMomentValue);

}

}

Output:

